Predicting Twitter Interactions in a Polarized Environment

Tommaso Bonetti *tbonetti@kth.se*
Hongjun Chi *hongjunc@kth.se*
August Paulsrud *augustpa@kth.se*
Donglin Wu *donglinw@kth.se*

June 9, 2023

Abstract

Link prediction algorithms are used to attempt to determine where new edges will form on graphs, and they boast a variety of applications spanning from social to biological networks. In this project, we adapt an existing link prediction model and apply it to a graph of tweets, with the goal being predicting interactions between Twitter users. We analyze five separate fake news stories which were spread on Twitter, using information about the tweets and the users that posted them to forecast which users will interact in a notoriously divisive environment. Our results show that the method yields a good prediction performance, being capable of correctly identifying the vast majority of interactions within very few guesses.

I. Introduction

Link prediction is one of the foremost problems in network theory, with well known applications in social, citation and biological networks, and more generally in recommender systems. Specifically, there is a substantial body of work concerned with predicting friendship links on different social networks, such as Facebook or Twitter. At the same time, the spread of fake news has become a very relevant issue, especially in the last decade. This has sparked a considerable amount of research on how to identify fake news stories and contrast their spread, and network theory has found its applications in this domain as well.

Combining link prediction with the study of fake news spreading could help identify and anticipate the posts users will interact with, giving social media platforms the possibility to moderate potentially harmful content by limiting the reach of posts that spread misinformation.

In this project, we propose and test a link prediction method that is specifically designed to predict interactions among Twitter users, as opposed to friendships (in Twitter's case, follows). The aim is to forecast which tweets, if any, a user will interact with, by retweeting, quote tweeting, or replying to them. The model is based on a proven algorithm, but a few modifications are made in order to fit the specific problem we want to address.

1.1 Challenges

Predicting Twitter interactions, specifically those in relation to fake news stories, poses specific challenges that might make it hard to apply standard link prediction techniques. In our analysis, we will consider tweets concerned with 5 different fake news stories, along with the friendship networks of their authors: these issues affect this data, as well as other collections of tweets regarding fake news. The dataset is described in more detail in Section 2.2.

First, interactions among users tend to be extremely sparse. By design, each tweet can reference at most one previous tweet, and it is not uncommon for tweets to be standalones with no references. If the tweets are represented as the nodes in a graph whose edges represent references (retweets, quote tweets, replies), this graph will always have $|E| \leq |N|$. Sparse networks are a known problem for link prediction algorithms, as the sizeable unbalance between positive and negative examples means that predicting the formation of no links at all is a better strategy than most others, reaching very high accuracy values in spite of being practically useless. Moreover, the extreme sparsity of an interaction network makes it challenging to even apply many proven link prediction algorithms, since in typical applications (social circles, co-authorship networks) there are no strict constraints on the number of edges each node can form. Our method, however, addresses both of these aspects, as will be discussed later.

In addition to this, fake news-related tweets (both those that spread fake news and those that reject them) are very likely to be published in an extremely polarized environment, where the social circles of the users who spread and reject misinformation tend to be highly clustered, with few interactions between them (see Section 2.2). This situation does not reflect the general landscape in which tweets garner interactions, where social circles tend to overlap more easily and be less polarized. Since our method is based on a supervised learning framework, however, we believe that training the model on the desired interaction network can yield good results both in this specific application and with general Twitter interactions.

1.2 Performance

After applying the necessary adaptations, our model showed good performance on different fake news stories, proving to have good generalization capabilities and a more than satisfactory accuracy.

The main drawback of our solution is the computational complexity of the training process: the efficiency of the model should be the first issue to be tackled by further research on this topic. We believe that the results we achieved are promising and that continuing to work on this problem has the potential to lead to interesting findings. An accurate interaction prediction model could be used to control the spread of fake news or to recommend relevant tweets to users, as well as other in fields outside of social media.

II. Related work

2.1 Random walks research

The proposed method uses the Random Walks with Restarts (RWR) algorithm, a popular graph theory algorithm widely used in information retrieval, recommender systems, and social network analysis [1]. RWR simulates random walks on a graph, where the walker moves to a neighbor node or restarts from the source node with a given probability. It can be employed to rank nodes by similarity to the source, identify communities, and detect anomalies in a graph. RWR handles both directed and undirected graphs. RWR is based on Markov chains, stochastic models that describe random events. It introduces a restart probability parameter to control the walker's restarts from a set of nodes. This ensures exploration of the entire graph, even if it's not connected.

In the context of fake news analysis, RWR identifies influential nodes that spread fake news and predicts nodes affected by it. This helps design countermeasures. The project applies RWR to model fake news spread by analyzing propagation patterns of fake and true tweets. By using RWR, it predicts likely links between source and previous tweets, moderating viral tweets during early stages.

Another method which inspired the project is the Supervised Random Walks (SRW) algorithm, proposed by L. Backstrom and J. Leskovec in their paper *Supervised Random Walks: Predicting and Recommending Links in Social Networks* [2]. The original method was designed to predict links on social and collaboration graphs, but we will be exploring its applicability to a interaction networks. The SRW method is based on a PageRank-like random walk that is biased towards visiting positive training examples more often than other nodes. This bias is learned in a supervised way, using a logistic regression model that takes into account both network structure and node and edge attributes. The resulting algorithm provides a principled way to combine rich node and edge features with network structure to make reliable link predictions without requiring tedious feature extraction or generation.

The optimization method chosen for the supervised learning task is dual annealing. This technique implies approximating the global minimum of the function to be minimized with Generalized Simulated Annealing (GSA) [3] and then enhancing the result by applying a gradient-based local search.

The 5 sets of Twitter stories that we will be using differ from the datasets used in [2] in multiple ways. For example, they are not social graphs or collaboration graphs, but rather collections of tweets related to specific topics or events. Additionally, the graph structure and node/edge attributes may be different from those found in social networks or collaboration graphs. To adapt the method for use on these datasets, we will explore various preprocessing techniques such as feature extraction and graph construction. We will also compare the performance of our adapted method with other graph models so as to provide a baseline for evaluation.

2.2 Twitter data

The data we have use in our project is a collection of graphs covering five fake news stories spread on Twitter, originally collected by A. Bodaghi and J. Oliveira for their paper *A study on real graphs of fake news spreading on Twitter* [4]. For each of the fake news stories, the dataset includes information to construct two graphs.

The first graph is the interaction graph, which consists of the tweets that are related to the news story in question. The tweets are labeled as either supporting the story, rejecting it, questioning it without holding a definitive position, or being unrelated. The nodes of the graph are connected by directed edges representing interactions, in the form of retweets, replies or quote tweets: the origin of each edge references the edge's destination. By the nature of the data, each node can have an out degree of 0 or 1, as each tweet can reference at most one other tweet. Some summary statistics about these graphs are presented in Table 1. Particularly, the time each tweet was posted is known and each of the tweets is linked to its author.

Story #	N	E	$\max k^{in}$	Δt
1	496	416	327	233 h
2	980	541	117	3 h
3	638	500	191	74 h
4	730	509	476	131 h
5	1199	1126	972	88 h

Table 1: Overview of the interaction graph for each story. N and E represent the number of nodes and edges, $\max k^{in}$ is the highest in-degree, and Δt represents the time elapsed between the publication of the first and last tweet in the graph.

The second graph is the user graph, which consists of the users, linked through directed edges to users they follow. This graph contains the authors of the tweets in the interaction graph along with their complete neighborhood. The nodes in the graph can then be other users who posted tweets contained in the interaction graph or unrelated users. In the latter case, no information about who the users follow or their amount of followers is provided. For each of the users who posted a tweet about the fake news story, the follower count, represented by the total in-degree on the user graph, is known.

The stories in the dataset cover contentious topics such as politics, religion and references to drug use in children's entertainment. Interactions between users might arise if they are in agreement about the topic or, less often, if they have a strong disagreement. This differs from the social networks the original method was developed for, in which all links are based on some form of mutual collaboration or friendship. Furthermore, the interconnections between users suggest a high clustering in the user graph, with a vast majority of connection being between two users that both endorse the fake news.

III. Method

The proposed model is a variant of the Supervised Random Walks algorithm [2] specifically adapted to suit this task. At the core of this algorithm is a biased random walk with restarts, where the probability of moving from one node to another is computed by means of a parametrized edge strength function. The parameters of this function are then optimized in a supervised learning task in order to ensure the best predictions.

3.1 Original problem formulation

Given a directed graph $G = (N, E)$, the problem is defined by a source node $s \in N$ and a set of edge candidates $C \subseteq N$. The edge candidates are divided into two disjoint subsets, the true destination nodes D and the no-link nodes L: D and L constitute, respectively, the positive and negative training examples. Each potential edge (u, v) is associated with a vector ψ_{uv} containing features related to the source and destination nodes and their relationship. The strength a_{uv} of each potential edge is computed as $a_{uv} = f_w(\psi_{uv})$.

3.2 Adaptation

Embedded user graph — An important element of novelty introduced in this project consists in the choice of the graph used to make predictions. The user graph encodes a lot of important information in the context of link prediction, and one could argue that interactions between users may even be forecasted based on user connections only. On the other hand, since the task is to predict links representing interactions between tweets, the interaction graph would be the obvious choice. An ideal strategy, then, would be one that can capitalize on as much information as possible, both from the user graph and the interaction graph: the approach adopted in this work is designed to do exactly that, training the model over a hybrid of the two.

From an abstract point of view the graphs are superimposed, and the interactions between tweets are modeled as edges between their authors. This effectively yields a directed multigraph, whose edges belong to one of two disjoint subsets:

- F, which is the original set of edges for the user graph and represents the *following* relationship: $(u, v) \in F$ if and only if user u follows user v.
- T, which is the set of edges for the tweet graph and represents the *interaction* relationship: $(u, v) \in T$ if and only if user u has published a tweet referencing a tweet from user v.

This conceptual view of the problem can then be simplified significantly through the choice of edge features. Since the interaction graph is much smaller and sparser than the user graph, with $|E_i| \leq |N_i|$ as opposed to $|E_u| = \mathcal{O}(|N_u| \log |N_u|)$ and $|N_i| \ll |N_u|$, the relevant information encoded in the user graph is simply embedded in the tweet graph by means of several edge features (see Section 4).

Ultimately, using the embedding allows to leverage valuable data concerning the social circle of each user and, at the same time, optimize the parameters of the edge strength function $f_w(\cdot)$ over the interaction graph, leading to a considerably lower computational cost for training the model. Performing random walks over the interaction graph, however, poses some challenges: in order to overcome them, the graph needs to be modified before training the model.

Modified interaction graph — Due to the strict constraints regarding edge formation among tweets, the interaction graph is disconnected and contains multiple sink nodes, producing a sub-stochastic block matrix on which a random walk will never converge to a meaningful score vector. For this reason, it is essential to apply some changes to G before fitting the model: this is the fundamental step to ensure that the SRW algorithm can be applied to Twitter interaction graphs. Let $\hat{G}_s = (N, \hat{E})$ be a modified version of G, where the nodes are the same and the edges are defined by the following criteria:

- Every edge in G becomes bi-directional: this improves the connectivity of the graph, eliminating sink nodes in the previously weakly connected component.
- The source node has outgoing edges to every node, including itself: since nodes in the interaction graph have at most one outgoing edge, a random walker will have no choice but to follow that edge, regardless of its strength. Adding edges from s to the rest of the graph, however, allows to bias the first step of the random walk in a meaningful way, ensuring a successful optimization and good prediction performance.

After the interaction graph has been modified appropriately, the edge candidates can be defined. For the purposes of this project, the set of candidates C will consist of the tweets that were published prior to the source tweet s, as well as s itself. Note that every node has one outgoing edge at most, which reduces the set of true destination nodes D to a single element d; additionally, for every node u with no outgoing edge, the true destination is defined as $d(u) = u$. This generalization facilitates solving the optimization problem, particularly the computation of the cost function, and it makes it possible to predict links by considering the top predictions overall, as well as by applying a threshold to the scores.

3.3 Random Walk with Restarts

Given the edge strengths a_{uv} and the modified graph $\hat{G}_s$, a first random walk transition probability matrix Q' is defined as

$$Q'_{uv} = \begin{cases} \frac{a_{uv}}{\sum_{z \in \mathcal{N}(u)} a_{uz}} & \text{if } (u,v) \in \hat{E} \\ 0 & \text{otherwise} \end{cases}.$$

The final random walk transition probability matrix Q is then computed by including the restart probability α for the source node s:

$$Q_{uv} = (1-\alpha) Q'_{uv} + \alpha \mathbf{1}\{v = s\}, \quad \alpha \in (0,1).$$

The stationary distribution p of the random walk with transition probability matrix Q yields a PageRank-like score for each of the candidate nodes, and the nodes with the highest scores are selected as the predicted link destinations.

3.4 Optimization

The goal of the supervised learning task consists in learning the parameter vector w for the edge strength function $f_w(\cdot)$ that yields the most accurate predictions. The task can be formulated as an optimization problem:

$$\begin{aligned} \min_{w} \quad & \|w\|^2 \\ \text{s.t.} \quad & p_l < p_d \quad \forall l \in L \end{aligned}$$

where p is the vector of PageRank scores, d is the destination node and L is the set of the no-link nodes. Minimizing the norm of w acts as a regularization measure, and the hard constraints require the score of the true destination node to be greater than that of every no-link node.

In the real case, however, it is highly unlikely to find a solution that satisfies all constraints. The original problem can then be relaxed using a similar technique to the hard-margin SVM relaxation, namely by adding a penalty for every violated constraint and removing constraints altogether. This relaxation gives the following formulation:

$$\min_{w} C(w) := \|w\|^2 + \lambda \sum_{l \in L} h(p_l - p_d),$$

where h is a differentiable function such that $h(x) = 0$ if $x \leq 0$ and $h(x) > 0$ if $x > 0$, and λ is a real-valued parameter regulating the trade-off between model complexity and goodness of fit.

The cost function $C(w)$ is then minimized using the SciPy implementation of dual annealing [5], an algorithm which combines Generalized Simulated Annealing (GSA) [3] with a local search over the accepted locations. There are two reasons that justify the choice of this algorithm over gradient descent:

- The distinctive ability of GSA to approximate a global minimum without getting stuck in local minima, which distinguishes it from most gradient-based minimization techniques.
- The gradient-based local search performed over the accepted locations, which further improves performance.

Although the optimization is mostly gradient-agnostic, the gradient of $C(w)$ still needs to be computed for the local search. However, as is well known, the score vector is the solution of $p = Q^T p$: this recursive relation makes it impossible to compute $\frac{\partial C(w)}{\partial w}$ in closed form. As a result, a power iterator that recursively applies the chain rule is used to approximate its value, computing successive estimates of $\frac{\partial p}{\partial w}$ and using the most recent one to update the estimate at the next iteration: the algorithm stops when both p and its gradient change by less than $\varepsilon = 10^{-8}$.

Finally, the algorithm can be extended by computing the cost function over multiple source nodes $s \in I$, even across different interaction graphs. The cumulative cost function is simply defined as

$$\tilde{C}(w) = \|w\|^2 + \lambda \sum_{s \in I} \sum_{l \in L(s)} h(p_l - p_{d(s)}),$$

and the gradient calculation is evidently independent for each instance $s \in I$. The ability to optimize the parameter vector w over multiple instances increases the generalization capabilities of the algorithm, as it reduces the likelihood of overfitting the training data.

IV. Experimental Setup

4.1 Data

The feature vector ψ_{uv} for the Twitter graph is designed so as to embed statistics from the user graph, as well as information about the authors a_u, a_v of both tweets in question and the interaction itself. Given the edge source u and edge destination v, the selected features are:

- The number of tweets posted by the author of the destination tweet.
- The number of followers for both authors, $k^{in}(a_u)$ and $k^{in}(a_v)$.
- The indicator of the event in which both tweets are from the same author, $\mathbf{1}\{a_u = a_v\}$.
- The number of interactions garnered by the destination tweet by the time the source tweet was published:
$$k^{in}(v)|_{t=t_u} = \left|\{(w,v) \in \hat{E} : t_w < t_u\}\right|.$$
- The time elapsed between the publication of source and destination, mapped on the $[0,1]$ interval using a sigmoid function:
$$\tau_{uv} = \frac{2}{1+\exp(\Delta t / c)}, \quad \Delta t > 0.$$
- The inverse of the length of the (directed) shortest path between the authors:
$$d_{uv} = \frac{1}{\text{dist}(a_u, a_v)} \cdot \mathbf{1}\{a_u \neq a_v\}.$$
- The Jaccard coefficient between the authors' neighborhoods:
$$J_{uv} = \frac{|\mathcal{N}(a_u) \cap \mathcal{N}(a_v)|}{|\mathcal{N}(a_u) \cup \mathcal{N}(a_v)|}.$$

4.2 Choice of functions and parameters

Loss function — The chosen loss function is the Wilcoxon-Mann-Whitney loss (WMW):

$$h(x) = \frac{1}{1+\exp(-\frac{x}{b})},$$

where $b = 10^{-6}$ is the width parameter. This function was preferred to the square hinge loss

$$h(x) = (\max\{0, x+k\})^2$$

due to its better performance. The reason for this is that the differences between PageRank scores $p_l - p_d$ are often very small: the squared hinge loss will either penalize negative differences in a similar way as positive differences or barely penalize positive differences, both of which are undesirable behaviours. On the other hand, a step-like function like WMW will penalize all and only positive differences by approximately the same amount, facilitating the optimization. This observation is backed up by the experimental evidence in [2], where WMW is noted to be the only function for which a reduction in the value of the loss corresponds to improved test performance.

Edge strength function — The selected edge strength function is instead the logistic function:

$$f_w(\psi) = \frac{1}{1+\exp(-\langle \psi, w \rangle)},$$

where the argument of the parametrized function is the inner product between the feature vector ψ and the learned weight parameter w. Using a function with bounded values results in better performance compared to unbounded functions such as $\exp(\langle \psi, w \rangle)$ or $(\max\{0, \langle \psi, w \rangle + k\})^2$, both of which tend to be highly biased towards nodes with a high in-degree.

Fixed parameters — The restart probability α is very important, since it regulates the expected number of steps taken by a random walker before restarting from the source node s. This parameter is particularly relevant for predicting links on the interaction graph: since this graph is not strongly connected, a random walk with restart probability $\alpha = 0$ will not converge to a meaningful stationary distribution.

The chosen value of was $\alpha = 0.3$, as it yields good validation performance both for Recall@K and AUC (see Figure 1) along with a quick convergence. It should be noted that $\alpha = 0.1$ gives similar results (even better in the case of AUC) but it slows down the training process, since a lower restart probability means a random walk will take longer to converge. As a result, the choice falls on the higher of the two candidates. The regularization parameter λ is instead set to 1, as suggested in [2].

4.3 Training data split

We split the interaction graphs into training, validation and test data using a 60/20/20 ratio over the number of nodes. The first 60% of the tweets constitute the training set, the subsequent 20% are used as the validation set and the last 20% are the test set. This approach can improve the reliability and reproducibility of results, while also keeping a rigorous distinction between training and test data.

As suggested by [6], we follow the temporal evolution of the interaction graphs, ensuring the test labels are hidden until the evaluation. In our case, keeping the clear distinction between training and test data means hiding the edges created by test tweets in every interaction graph from the test set.

The validation set is an important component of the supervised learning workflow. It helps prevent overfitting by evaluating the model's performance on data that it has not seen before, and it can be used to perform hyperparameter optimization. In our project, we use the validation data to evaluate the impact of different values for the restart probability α on the accuracy of the model, then we pick the value that gives the best validation performance and use it to make predictions on the test set.

V. Evaluation

After training the model, the estimated weight vector w is used to compute edge strengths a_{uv} and run a random walk with restarts for every source node in the test set I. For a given source node s, the resulting stochastic score vector p gives a proximity measure between s and every node $v \in N$, including itself: these scores will be used to predict the most likely links.

5.1 Performance metrics

The predicted links are evaluated using two metrics: the area under the ROC curve (AUC) and the recall at top K (Recall@K), for $K \in \{1, ..., 10\}$.

Particularly, given a set of source node instances I, the recall at top K is defined as:

$$\text{Recall@K} = \frac{1}{|I|} \sum_{s \in I} \frac{TP_s^{(K)}}{TP_s + FN_s},$$

where $TP_s^{(K)}$ is the number of true destination nodes for s in the top K predictions (i.e., the nodes with the highest scores in p) and the denominator is the total number of true destination nodes for s.

Notice that, in our case, $TP_s + FN_s = 1$ for every instance (each node can only have one true destination), and $TP_s^{(K)}$ is simply an indicator of whether the true destination node is part of the top K predictions: as a result, Recall@K is the fraction of instances whose true destination node is in the top K predictions.

5.2 Comparison models

The predictions of our model are compared against those made by a few baseline models. These simple and well known models were added to our evaluation to provide a point of comparison for the performance, and will be introduced in this section.

Random attachment — Random attachment decides which node to connect to uniformly at random, each node having an equal probability to be picked. For our experiments, the node that is forming the link is also a candidate, a self link being interpreted as no reference to other tweets. To get the top 10 predictions, the random attachment algorithm is executed until 10 unique nodes have been selected. If the graph contains less than 10 nodes, it is executed until all the nodes have been selected.

The probability that a link will be formed with a node n_l is P_l, defined as

$$P_l = \frac{1}{|N|}$$

where $|N|$ is the number of nodes in the graph.

Preferential attachment — Preferential attachment decides which node to connect to based on the nodes' degrees, each node having a probability to be picked proportional to its in-degree. To get the top 10 predictions, the preferential attachment algorithm is executed until 10 unique nodes are picked. If the graph contains less than 10 nodes, it is run until all the nodes have been picked.

The probability that a link will be formed with a node n_l is P_l, defined as

$$P_l = \frac{k^{in}(n_l) + 1}{\sum_{n_s \in N}(k^{in}(n_s) + 1)}$$

where N is the set of nodes in the graph and $k^{in}(n_s)$ is the in degree for node n_s.

Ordered top attachment — Ordered top attachment decides which node to connect to by ordering all nodes in the graph by descending current in-degree and selecting the highest ranked node. To get the top 10 predictions, the top 10 nodes are selected. If the graph contains less than 10 nodes, all of them will be selected.

The node that a link will be formed with, n_l is selected by

$$n_l = \arg\max_{n_s \in N} k^{in}(n_s)$$

where N is the set of nodes in the graph and $k^{in}(n_s)$ is the in degree to node n_s.

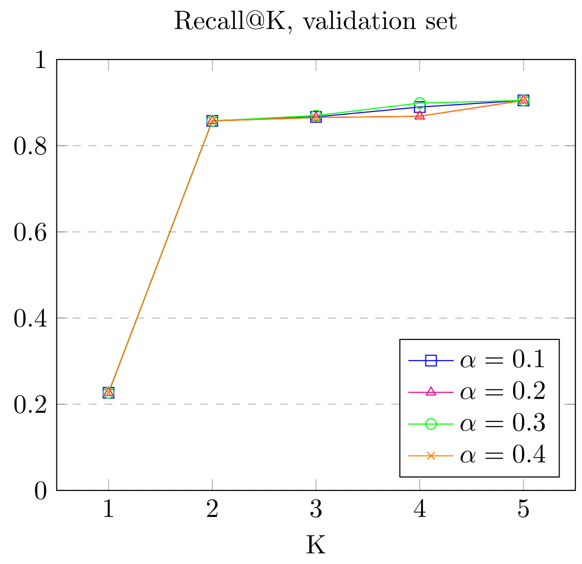
VI. Results

In this section we present the results we achieved. First we show the performance of our model on the test data, and then compare this performance to that of the simpler prediction methods.

Figure 1 shows the performance on the validation set with different values of α, whereas Figure 2 displays the Recall@K metric achieved by our model on the test set. The AUC on the test set is 0.942.

On the other hand, Figure 3 shows the performance metrics obtained when applying our model to each specific fake news story. It is interesting to notice how the performance varies between graphs: this is mainly due to the different number of examples for the news stories. The smallest graphs will weigh less on the cost function, hence the predictions on them will be less accurate than on the others: in fact, the smallest interactions graphs (1 and 3) yield the worst performance under both metrics.

Lastly, Figure 4 compares the test performance of the baseline models to that of our SRW-based model. Unsurprisingly, these very simple models fail to capture the complexity of the interaction distribution. Nonetheless, the results obtained by ordered top attachment and preferential attachment uphold the intuition that the in-degree of a tweet, i.e. the number of times it is referenced, has a positive influence on the likelihood of edge formation. The AUC is omitted for this comparison, since we deemed Recall@K to be more representative of the performance of each algorithm due to the high imbalance between positive and negative examples.



α	AUC
0.1	0.966
0.2	0.960
0.3	0.963
0.4	0.960

Figure 1: Performance on the validation set for different values of α.

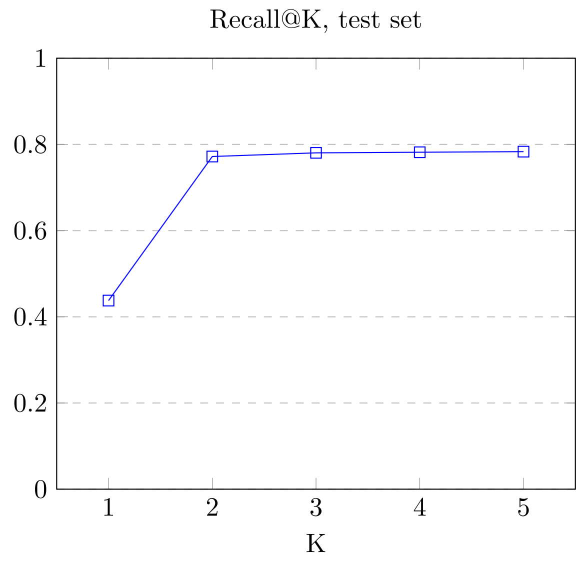
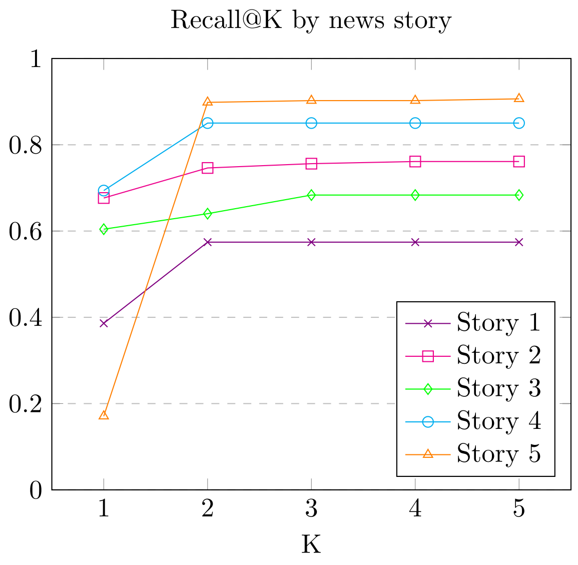


Figure 2: Recall@K on the test set.

VII. Discussion

The results clearly show that our SRW-based model can be used to effectively predict edges on the interaction graphs. This is especially true if it is used in a setting,



Story	AUC
1	0.845
2	0.909
3	0.866
4	0.942
5	0.964

Figure 3: Performance on the test set, evaluated separately for each fake news story.

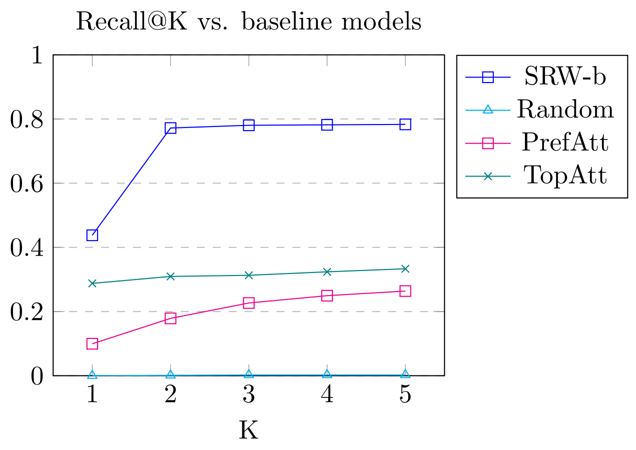


Figure 4: Recall@K on the test set for our model and the baseline models: random attachment, preferential attachment, ordered top attachment.

such as generating recommendations, where false positives are not a critical issue: in fact, as shown above, the top 2 predictions generally need to be considered in order to correctly identify the true edge destination in the vast majority of the cases.

7.1 Performance on validation data

A notable result when predicting links on the validation data is that the second guess has a higher probability of being correct than the first, a behaviour that is not present when predicting links on the test data. This may be caused by some underlying difference in the data, and it seems to suggest a slight shift in the interaction distribution as time progresses, but it is hard to gauge whether this is actually the case.

The restart probability α, for the values tested, has little impact on the performance of the model. The difference is most distinguishable in the AUC result, where it still is less than half of a percentage point with $\alpha = 0.1$ being the best performing parameter value, while $\alpha = 0.3$ performs better by a small margin with respect to Recall@K. This result is somewhat surprising, as the number of expected random walk steps before restarting vary significantly as α increases. By observing the learned parameter w, however, it is apparent that the weights are not particularly similar across different restart probability values: this would suggest that the structure of the graph has a much higher influence on the result than the edge strengths. Given how sparse the interaction graph is, it is not difficult to see why this would be the case: after the first hop, a random walker will have very few options as the next destinations, resulting in the same nodes being visited very often regardless of the restart probability.

7.2 Performance on test data

The performance of the model on the test data is mostly in line with that on the evaluation data. The only significant difference is that Recall@1 is significantly higher on the test data than the validation data, resulting in almost twice the rate of correct predictions. This gap, though, is immediately closed at the second prediction, where Recall@2 takes similar values in both cases. The cause of this may simply be the that the edges formed by nodes in the test set are formed in a similar fashion to those in the training set, possibly more so than those in the evaluation set. It could also be possible that the increase in performance is due to more information being available in the graph, but ultimately, more testing would be needed to explain this behaviour.

The Recall@K values also clearly hit a plateau after $K = 2$. By analyzing the recall achieved by different nodes based on their true destination, the reason for this sudden stabilization seems fairly clear: our model is strongly biased towards predicting self-loops (i.e., standalone tweets) or edges to the biggest attractors (i.e., the tweets that already have many interactions). This bias is intrinsic to our algorithm, and more specifically to the random walk with restarts at its core. When dealing with networks as sparse as our interaction graphs, where even after various adaptations $|E| = \mathcal{O}(|N|)$, it is unsurprising that a random walker will visit the source node and the nodes with the highest in-degree much more likely than any other.

Specifically, splitting the recall based on the destination node shows that only standalone tweets are identified correctly with the first prediction at an almost perfect rate, while references to big attractors are identified within the first two predictions about 75% of the time and other references are hardly ever guessed even in the top 10. This shows that standalone tweets and references to big attractors are by far the most common types of interactions, and the good performance of our model is mainly due to its ability to correctly identift them; an interesting future research direction, however, would be developing a model that is capable of effectively predicting references tweets with very few, if any, previous interactions.

7.3 Comparison to simple models

All the baseline methods considered performed worse than our model. The performance of the random model is highly dependent on the size of the graph, and as the test set contained the interaction graphs at their largest it performed poorly. The ordered top attachment method, which could not predict self edges and thus could not correctly handle any tweet which did not reference a previous tweet, still performed the best of the comparison models. This shows the importance of the in-degree of existing nodes when predicting new links.

This is further highlighted by the recall of almost 30% for the first prediction, indicating that almost one out of three tweets will form an edge towards the node with the highest in-degree. Nevertheless, the gap in performance between the ordered top attachment model and the SRW-based model shows that other factors are also relevant in predicting interactions, and that our method can leverage their importance to attain a superior result.

One significant benefit the baseline models offer over ours graphs is their complexity: especially the random attachment and top list attachment methods are superior in this regard as their complexity, unlike that of our method, is not dependent on the size of the graph.

7.4 Choice of test data

One factor which may have impacted our results is the choice of the test data. Splitting the data with a 60/20/20 ratio, based on the time the edge was added, is meant to ensure that the temporal evolution of the data is preserved, and also to provide transferability to real-world scenarios, where predictive methods are deployed after training and must deal with unlabeled data. This approach, strongly suggested by [6], was applied successfully, but has left us with a few considerations.

The test data was collected from the last 20% of the nodes added to the graph. These last nodes may have behaved differently from the previous nodes, which could have impacted our result, especially since our model is fairly simple and might not have been able to capture the full complexity of the phenomenon of Twitter interactions. Whether or not this is the case could be tested by scrambling the order of the data before splitting it; however, this would mean that some information that is hidden when testing the model would be used in the training. Ultimately, though, applying this rigorous distinction between training and test data is the only way to provide an accurate evaluation of the performance of our model in a setting that closely mimics a real-world application.

7.5 Model complexity

The main drawback of our method is its complexity. As good as it is for prediction performance, the supervised learning framework requires computing a random walk with restarts and approximating the gradient of the cost function via power iteration once for every source node and for every iteration during optimization. This clearly has a high computational cost, both in terms of time and memory. Training our model on around 2,500 instances took more than 24 hours, although this runtime could be drastically improved by using high-performance computers and parallel computation, which would allow to leverage the independence on the cost function across different instances. As explained in [2], the complexity of this model is related to the size of the graph, but also to the number of features and to the restart probability: for this reason, we elected to only use 8 edge features and we preferred higher values of α (see Section 4.2).

VIII. Conclusion

Our results clearly show that the adapted SRW method yields a good accuracy when applied to the contentious environment of the Twitter discussion regarding fake news. This can be seen in the comparison to the baseline link prediction models, which all showed far inferior results. A remaining issue is the methods high computational cost, which could limit its use in some time-sensitive applications.

There exists plenty of opportunity for future research on a SRW-based algorithm tailored for interaction prediction. Decreasing the complexity would be the foremost concern, possibly by parallelizing the code or by reducing the number of features considered, followed by developing a variant of this model that is capable of predicting interactions with non-attractor tweets. Research could also be performed on other contexts or environments in which the method can be used.

References

[1] H. Tong, C. Faloutsos, and J.-y. Pan, "Fast random walk with restart and its applications," in *Sixth International Conference on Data Mining (ICDM'06)*, 2006, pp. 613–622. DOI: `10.1109/ICDM.2006.70`.

[2] L. Backstrom and J. Leskovec, "Supervised Random Walks: Predicting and recommending links in social networks," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, ser. WSDM '11, Hong Kong, China: Association for Computing Machinery, 2011, pp. 635–644, ISBN: 9781450304931.

[3] Y. Xiang, D. Sun, W. Fan, and X. Gong, "Generalized simulated annealing algorithm and its application to the Thomson model," *Physics Letters A*, vol. 233, no. 3, pp. 216–220, 1997, ISSN: 0375-9601.

[4] A. Bodaghi and J. Oliveira, "The theater of fake news spreading, who plays which role? A study on real graphs of spreading on Twitter," *Expert Systems with Applications*, vol. 189, p. 116 110, 2022, ISSN: 0957-4174.

[5] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: `10.1038/s41592-019-0686-2`.

[6] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 751–782, Oct. 2014.